<center>REMARKS</center>

Applicant believes that the Patent Office is mistaken regarding the nature and scope of applicant's invention and the cited prior art, and for some claims, has failed to establish a *prima facie* case for obviousness. First and foremost, neither of the cited WebWriter articles describes or suggests running edited applications during editing. The WebWriter Editor displays dynamic parts of pages only as placeholders during editing, not as functional content. This means that, in general, pages have a different appearance during editing and at run-time.

Applicant's claims recite these and other distinctions over the cited combination in various ways, as further explained below. Applicant has amended certain claims to clarify these distinctions, and Applicant submits that all pending claims are patentable over the cited references for the reasons detailed below.

I. GROUNDS OF REJECTION TO BE REVIEWED

   A. Whether claims 1-6, 8, 22-33, 41-43, 51-96 and 114-128 are unpatentable under Section 103(a) over the combination of the 1996 article by Crespo and Bier entitled: *WebWriter: A Browser-Based Editor for Constructing Web Applications* ("WebWriter I") and the 1997 article by Crespo, Chang and Bier entitled *Responsive Interaction for a Large Web Application: The Meteor Shower Architecture in the WebWriter II Editor* ("WebWriter II").

II. ARGUMENTS

A. THE EXAMINER IS MISTAKEN REGARDING THE NATURE AND SCOPE OF THE CITED PRIOR ART

   i.   Overview

The Patent Office asserts that the combination of the WebWriter I article and the WebWriter II article makes Applicant's claims obvious. However, Applicant respectfully disagrees, and believes that the Patent Office is mistaken regarding the nature and scope of the disclosure in the WebWriter articles.

The editor described in WebWriter I is mainly concerned with editing the static parts of a page, and shows dynamic parts as placeholders during editing. For example, the WebWriter I article states that "*that static parts of each page are created in advance by the designer using a text editor*" (*see* p. 2, 2nd para. under heading "The WebWriter Application Model") and further,

"*WebWriter then adds a placeholder for the output area [which] will be replaced **at runtime** by the output of a script . . . .*" (*see* page 4, 2nd para. under heading "Placing Dynamic Areas", emphasis added). In contrast, Applicant's invention is concerned with editing components, which are dynamic parts. Further, Applicant's invention shows the executed dynamic parts during editing, and except for the addition of editing features, such as e.g. handles, the appearance and function during editing is similar as at run-time. This is achieved by running the edited web application during editing. In contrast, WebWriter I runs the edited application only after editing, as discussed further below, and makes no teaching or suggestion of running an application during an editing operation.

Applicant's Claims express these distinctions by stating, for example, that (1) the edited application is running during editing, or (2) the component displayed during editing has a similar appearance as on the generated page with the addition of editing features or (3) components cooperate with the editor during editing.

For example, claims 1 and 59 require a document generator that runs the application during editing, claim 90 requires that scripts in the edited document remain running during editing, and method claim 125 explicitly requires a running step during editing. Claim 26 requires that the documents be similar at run time and during editing, to wit: "*at least a part of the second document appears and functions similar to the first document*". Claims 74 and 114 require components that cooperate with the editor during editing.

Section ii below contains a more detailed discussion explaining why the WebWriter articles do not teach or suggest running applications during editing. Section iii discusses the internal architecture of WebWriter and explains why it is not even possible or feasible to run the application during editing when using that architecture. Section viii discusses components cooperating with the editor during editing.

ii.   The Examiner Misinterprets WebWriter

In the current Rejection, the Examiner asserts that WebWriter I is capable running the edited web application while editing, which is incorrect. In fact, WebWriter I follows a strict separation of editing time and run time. During editing, WebWriter I only shows the static parts of a template page while dynamically generated parts are displayed as placeholders. Upon completion of editing, the template page is saved. At runtime, the dynamically generated parts

are executed and their output is displayed. This is explained as noted in Section i above, and also on p. 6, col. 2 of WebWriter I, in the section entitled "Running the Application:" "*Once a Web Writer application is created and **saved** to disk it can be **run** in one or two ways*" (emphasis added)..

The output generated by scripts in general looks very different than the placeholder, so that Pages displayed by WebWriter during editing and at runtime usually look very different as well. For example, Fig. 7(b) of the WebWriter I reference shows the editing view and Fig. 6(b) shows the runtime view of a single page template implementing a file listing application. The runtime view correctly shows a list of files, while the editing view shows a placeholder consisting of handles surrounding the output formatting specification *"[!Line]"*. Also, Fig. 10(b) and Fig. 11 show the difference between the runtime and editing views in WebWriter I of a page template implementing a walkie talkie application. The runtime view in Fig. 10(b) shows the text of a message received by the walkie talkie while the editing view just shows the placeholder, namely *"[!Line]"* surrounded by handles.

On page 8 column 2 and page 9 of the WebWriter I reference, there is a main section labelled "Implementation" split into two subsections "The WebWriter Editor" and "The WebWriter Page Generator". The editor and page generator are separate programs, the editor being used during editing and the page generator being used at run time. The first paragraph of the "WebWriter Page Generator" section on page 9 explains that the Web Writer Page Generator is used at run-time, i.e. while an application created with WebWriter is running: "**When a WebWriter application is running**, each new page is assembled by the **WebWriter Page Generator**, **another** server based CGI C++ **program**..." (emphasis added) .

The reasoning employed by the examiner in rejecting applicant's independent claims mainly refers to this "Web Writer Page Generator" section on page 9. However, since the Web Writer Page Generator is a separate program used at run time only after completing an editing task, it is in fact irrelevant to the claimed editor features. The examiner specifically mentions the phrase "on the fly," but this phrase does not mean what the examiner suggests. On page 9 column 1, the WebWriter I article states: "*The Page Generator generates a new page on the fly as it scans through the template page,*" and this just means that the WebWriter Page Generator is implemented using the "on the fly" technique, which is a common technique to implement

parsers. The sentence therefore just explains the inner workings of the WebWriter Page Generator, but does not express in any way that the page generator is running during editing.

With respect to WebWriter II, the examiner only refers to it for establishing prior art for "server" or "web server" on page 3 of the recent office action. The examiner does not suggest that WebWriter II discloses running dynamic web run applications during editing.

iii.   Editor Architecture

The WebWriter and applicant's editor have very different architecture. As a classical editor WebWriter I and WebWriter II have load and save operations. When loading a document, it is transformed into a content tree. Editing operations modify the tree. Finally, on a save the tree is unparsed and stored into a file. During editing after each operation, the tree is traversed and displayed to the user. (See page 8 column 2 Headline The WebWriter Editor). In addition, there is the WebWriter Page Generator that, after the edited page has been saved, reads a file and executes scripts. This classical architecture requires that applications be executed only when editing is finished and the tree is sent to the server and saved to a file, because the file is needed for execution. During editing, documents are displayed inside the editor, i.e. inside an environment that considerably differs from the runtime environment, e.g. URLs, URL parameters, and the page generator differ. In order to successfully run an application, however, it must run in environment that closely resembles the runtime environment.

In contrast, applicant's editor runs the application being edited in a first browser window and the client part of the editor in another window. The user interacts with the application running in the first window as usual. It runs in its normal runtime environment, e.g. using its normal URLs, URL parameters etc. and also uses its normal page generator. Therefore, it appears fully functional. While editing, the page generator adds additional editing features to the generated pages, in a way not interfering with the normal operation of the application. The editing features include handles. The user can click on a handle or interact with the editor window to issue a modification. Then the client and server part of the editor perform the modification directly in the document template file. Afterwards the editor asks the browser to reload the page in the first window, which makes the now modified application run by the page generator as usual.

iv.   The Combination of WebWriter I and WebWriter II Is Improper

In rejecting applicant's claims, the examiner has relied on two prior art references describing two variants of the WebWriter Editor referred to as WebWriter I and WebWriter II. WebWriter II is a newer development from the same authors trying to solve the same problems as WebWriter I, so one might assume that WebWriter II already contains as far as possible all the features of WebWriter I.

WebWriter II, however, uses a different architecture called "meteor shower" which performs most of the editing operations on the client computer (except for loading the document before editing and saving the document after editing), while WebWriter I did most of the editing operations on the server. (*See* WebWriter II, sec. 1.2 entitled "Increasing interactive performance"). This new architecture makes WebWriter II work faster but teaches away from applicant's idea because running the edited server side application necessarily takes place on the server. Applicant's editor does the editing in a distributed way, many operations are done client side (e.g. displaying information about components and running client side parts of an edited document during editing) while others are done server side (e.g. modifying the edited document and running the server side parts of it during editing).

Because of the different architectures used in WebWriter I and WebWriter II applicant submits that it is not obvious to simply combine WebWriter I and WebWriter II as suggested by the Examiner. Including a feature from one version into the other requires adaptation to another architecture (e.g. moved from server to client or vice versa), which may not be trivial or even impossible. To pick and choose features from different prior art disclosures in order to form an obviousness rejection may amount to hindsight reasoning, which is impermissible. Applicant therefore submits that the combination is improper, and on that basis, applicant submits that the Examiner has failed to establish a *prima facie* case of obviousness.

v.   The Examiner Fails to Provide Reasoning for All Independent Claims

On page 3 of the current action, the Examiner lumps all of the independent claims together, then describes his assertion that WebWriter I discloses a page generator program and an editor program as claimed. However, while that basis for rejection may be applicable to independent claims 1 and 59, the language and structure of independent claims 6, 22, 26, 51, 74, 90, 114 and 125 are much different than that of claims 1 and 59. Further, the Examiner did not

consider any differences in claim language or scope, nor did the Examiner provide any detailed reasoning that would support rejection of these claims of different language and scope, and therefore, the Examiner has failed to make a prima facie case for obviousness of independent claims 6, 22, 26, 51, 74, 90, 114 and 125.

vi.   Components

Applicant's invention allows the developer create web applications by plugging together software components on document templates using the inventive editor. These special type of components contain instructions, are executable on the server side, and dynamically generate browser code. In applicant's preferred embodiment, components are implemented as objects of an object oriented programming language.

Unfortunately, the current rejection does not specify what is considered prior art for components. With respect to claim 2, the first claim mentioning the terms components and document templates, the examiner refers to page 2, column 1 of WebWriter I, see Heading "*The WebWriter Application Model*" and refers to the term "*template page*". Therefore, applicant assumes that WebWriter's template pages are alleged to be prior art for the claimed page templates. The WebWriter I article defines dynamic areas as locations within a template page ("*... template page, which includes the specification of locations within the page, called dynamic areas*" ...) and adds that a script is specified for each dynamic area. So applicant assumes that the examiner interprets scripts specified for the dynamic areas as prior art for applicants components.

The cited section also states that the scripts are run at run-time to provide computed content for the dynamic areas ("*dynamic areas, where computed content should be inserted at run-time*"... "*a script that will be run to provide to provide the computed content for that dynamic area*" (emphasis added)).

vii.   Components may Cooperate with the Editor

Just as for WebWriter's scripts of dynamic areas, Applicant's components are able to run at run-time. However, applicant's components are run and used during editing as well, and this feature is not taught or suggested in the cited combination. In fact, applicant's components cooperate with the editor during editing in various ways, e.g. by generating browser code for

display of the component inside the editor, by drawing handles, or by collecting and providing information about the component to the editor.

In contrast, the scripts specified for WebWriter's dynamic areas are not used during editing, but just replaced by placeholders. ("*WebWriter then adds a **placeholder** for the output area at the current insert location point*")

In fact, applicant submits that a further distinctive feature is "components cooperating with the editor during editing." The present Office Action does not cite any specific portion of WebWriter as disclosing this feature, because of course it is not disclosed. WebWriter's dynamic areas call independent unix scripts at runtime, as explained on p. 9 at the end of column 1 and top of column 2: "*When WebWriter encounters an OUTPUT tag it ... runs the specified program on its arguments ...*". The WebWriter Editor cannot be extended by any external components. This is explained in WebWriter I on page 5 column 1 Section entitled "Script" "*The user selects from a small set of **built-in** modules ...*" (emphasis added). It is only the WebWriter Page generator that can call arbitrary scripts and more specifically scripts of dynamic areas at run time.

This distinction is recited in the claim language of claim 74 "*the components having the ability to cooperate with the editor*" of claim 114 "*the components including first features adapted to cooperate with an editor in dynamically editing said component.*"

viii.   Nested Components

WebWriter's dynamic areas are represented using a special <output> tag as explained on page 9 column 1. The output tag contains a formatting string. As explained at the end of column 1 and top of column 2 of page 9 "*When WebWriter encounters an OUTPUT tag it extracts the SCRIPT field and formatting string, runs the specified program on its arguments, formats the results using the formatting string and writes the result to the output.*". There is no teaching or motivation that the formatting string could contain itself nested OUTPUT tags.

In contrast, applicant's components can be nested. Inside a component on a document template, HTML code and other components can be denoted. An enclosing component can generate browser code for insertion at the begin tag and the end tag and it can also dynamically control insertion of content into the final document. For example, the component can hide its content, insert it once, or multiple times. Thus, the components nested inside may dynamically

be hidden, or multiple instances of a component may be displayed. In this way, the number and kind of components on a generated page is no longer static but can dynamically change at run time.

This is denoted using the claim language of independent claim 74: "*the set of components on the generated documents can vary for different document requests for said document template;*" and, for example, dependent claim 4: "*wherein at least one of the components contains at least one other component;*" and finally claim 8: "*a component is nested within a component*".

Note that not only do applicant's components have these features, but also applicant's editor has the ability to handle components with these features and, for example, has operations to place one component inside another one, or display multiple instances of a component during editing.

ix.   Components Interactively react on User Input

Some of Applicant's components can not only display themselves during a first browser request, but also on a subsequent request, after user interaction, process the data sent back from the browser to the server. These components are called interactive. Several dependent claims are based on interactive components.

In contrast, the scripts called by WebWriter's dynamic areas are just used for processing of a single request. As explained on p. 9 of WebWriter I, left column, Heading The Web Writer Page Generator, the scripts are called during document generation to produce output for an output area. This is taking place during processing of a single request. The scripts do not have a concept of multiple requests or subsequent requests and can therefore also not react in any way on a subsequent request.

This feature is recited for example in claim 24:"*at least one of the components ... can react on subsequent document requests containing user responses by executing selected instructions of said component.*"

B.   THE INDEPENDENT CLAIMS ARE PATENTABLE OVER THE COMBINATION
      OF WEBWRITER I AND WEBWRITER II

Claim 1

Claim 1 is an independent claim that is directed to a basic two-part structure: a document generator that runs a web application and an editor. It requires including editing features in generated documents and the editor program operating via the editing features. The claim does not rely on components.

As explained in section A.ii the WebWriter Editor does not run the edited application during editing. In stark contrast, applicant's editor does. Claim 1 is considered patentable over the cited combination at least because (1) the recited document generator runs at least part of the edited application by generating documents that include editing features, and (2) the recited editor operates via editing features incorporated into said generated documents. This is recited in Claim 1 as:

> a document generator program running at least part of one of the applications
> being edited and generating the generated documents, said generated documents
> including additional editing features for interpretation by the browser program; and
>   an editor program dynamically operating on the generated documents displayed
> by the browser program via the editing features.

The recent office action refers generally to the description of the WebWriter Page Generator in the WebWriter I article (see Office Action dated Sep. 17, 2008 at p. 3), but provides no specific reasoning to support the rejection. There is no teaching or suggestion in either WebWriter article that the WebWriter Page Generator inserts any editing features into the generated pages. In fact, this would not make any sense because the WebWriter Page Generator does not run during editing – it runs only after editing, as explained in section A.ii. The "on the fly" phrase used in the prior art article and specifically cited by the examiner just specifies the internal working of the page generator, e.g., at runtime, the placeholders are replaced with the dynamic content, and the article does not teach or suggest running the WebWriter Page Generator during editing.

In addition, the phrases "editor program operating on the generated documents" and "via the editing features" are key structural limitations of the claim language that were not specifically addressed by the Examiner, and which connect the running application to the editing process. The claim recites that the editing features are part of the generated documents, and the

preamble states that the generated documents are displayed by the browser. The editor is then coupled via the editing features, and *voila*, you are editing a running application in the browser.

The examiner cited WebWriter II only to incorporate the explicit teaching of server computer and web server. This, however, has no influence on the working of the WebWriter Page Generator and thus, Claim 1 is patentable over the cited combination.

Claim 6
As explained in section A.ii and A.i, WebWriter replaces dynamic parts of pages with placeholders during editing. The placeholders show the script name and a formatting string and so usually look very different from the appearance of a component on a generated page. In contrast, claim 6 has been amended to require that the components have a similar appearance as on the generated page. Therefore, applicant submits that the claim is patentable over the cited combination.

The examiner did not include specific reasoning for the rejection of claim 6, but instead relied upon his reasoning as applied to claim 1. However, Applicant submits that this reasoning is simply not applicable to claim 6, in part, because claim 6 recites components. Further, the failure to provide detailed reasoning amounts to a failure to make a *prima facie* case of obviousness.

Claim 22
Applicant amended claim 22 to clarify that the editor indeed operates on the documents generated by the page generator and by executing the components.

As explained in section A.ii the WebWriter Editor does not execute part of the edited application during editing. As described in WebWriter I page 9 the WebWriter scripts of dynamic areas are executed only after editing by the WebWriter Page Generator. In contrast claim 22 requires an editor operating on documents generated by the document generator that has instructions for executing components and so requires that the components are executed during editing. Therefore applicant submits that the claim is patentable over the cited combination.

The examiner did not include specific reasoning to support his rejection of claim 22, but relied on his reasoning as applied to claim 1. Applicant submits that this reasoning is not applicable because the claim has been amended as shown above, and also because claim 22

includes the recitation of components while claim 1 does not. Since the Examiner has failed to provide detailed reasoning, he has not made a *prima facie* case of obviousness.

Claim 26
Claim 26 describes a system to modify a first document stored on a server whereby a second document is displayed to the user that appears and **functions** similar to the first document. Applicant emphasizes the fact the recited elements of the claim are located on the server, wherein the server comprises:

> a document store;
>
> a first software program including instructions for transforming at least one first document into a second document having features which permit editing of the first document such that at least a part of the second document appears and functions the same as the first document; and
>
> "*a second software program including . . .* instructions *to modify the first document,*

Claim 26 stands rejected on the basis of a combination of WebWriter I and WebWriter II. As explained in detail in section A.ii WebWriter does not execute the application being edited during editing. Therefore it is clear that the application being editing does not function at all during editing. In contrast the claim requires "*at least a part of the second document appears and* ***functions*** *similar to the first document.*"

In fact WebWriter transforms a first document being edited into a second document which permits editing of the first, thereby trying maintain the appearance of the static parts of the document. However, as discussed in section A.i, WebWriter does not make any effort to maintain the appearance of the dynamically generated parts of the document or of the functionality of the document.

The examiner did not include specific reasoning to support his rejection of claim 26, but relied on his reasoning as applied to claim 1. Because the Examiner has failed to provide detailed reasoning, he has not made a *prima facie* case for obviousness.

Therefore, Claim 26 is patentable over the cited combination.

Claim 51
Claim 51 is an independent claim describing a system for editing components on web document templates. As explained in section A.vii, the WebWriter Editor does not use the scripts associated with the dynamic areas. These scripts are used only after editing, at run-time. In contrast, claim 51 has been amended to require components having features to cooperate in

<u>editing the component</u>. Therefore, applicant submits that the claim is patentable over the cited combination.

The examiner did not include specific reasoning for claim 51 but relied on his reasoning of claim 1. Applicant submits that this reasoning is not applicable because the claim has been amended as shown above, and also because claim 51 includes the recitation of components while claim 1 does not. Because the Examiner has failed to provide detailed reasoning, he has not made a *prima facie* case for obviousness.

<u>Claim 59</u>
Claim 59 is an independent claim directed to a software development system for dynamic web documents.

As explained above, the inventive editor is capable of running a dynamic web document being developed during editing and so the dynamic web document appears functional during editing. This distinction is expressed by the following claim language, "*generated documents ... which look and function similar to the end user's view of the documents*" and by *"the editor program comprising first instructions for requesting the document generator to process a dynamic web document leading to a <u>generated document</u>.*" This language makes clear that the editor indeed runs the dynamic web document during editing, by requesting the document generator to process a dynamic web document. Additional claim language, namely "*instructions to modify **the dynamic web document***" make clear that the dynamic web document that is being edited is the one being requested.

The examiner did not include specific reasoning for claim 59 but relied on his reasoning as applied to claim 1, wherein the examiner refers to the WebWriter Page Generator. The WebWriter Page Generator, however, runs after editing as previously explained, and so it is clear that the WebWriter Page Generator does not receive requests from the WebWriter Editor during editing. In contrast, the claim requires *"the editor program comprising first instructions for requesting the document generator to process a dynamic web document leading to a generated document.*" For all these reasons, applicant submits that claim 59 is patentable over the cited references.

<u>Claim 74</u>

Claim 74 is an independent claim describing a software development system for document templates and stands rejected as obvious based on the combination of WebWriter I and WebWriter II. In the Rejection, the Examiner did not include specific reasoning for claim 74 but handled claim 74 together with claim 1. Because the Examiner has failed to provide detailed reasoning, he has not made a *prima facie* case for obviousness.

Nevertheless claim 74 requires "*wherein the set of components on the generated documents can vary for different document requests*" and "*the components having the ability to cooperate with the editor*". Claim 1 does not require "components". Therefore, applicant submits that the examiners reasoning given is not applicable and therefore irrelevant for claim 74. As explained in sections A.vii and A.vi, WebWriter I and WebWriter II do not teach or suggest components cooperating with the editor. In contrast, claim 74 explicitly requires "*the components having the ability to cooperate with the editor*".

The claim explicitly requires that the set of components on the generated documents can vary for different document requests for the same document template. In WebWriter, each dynamic area is filled by its script exactly once per request, as explained in section "The Web Writer Page Generator" on page 9 of the WebWriter I article. It is not possible that a script associated with a dynamic area is excluded from generation or generated multiple times and so the number of scripts of dynamic areas executed per template page is fixed. In contrast, the claim requires that the set of components on the generated documents can vary for different document requests. For all these reasons, applicant submits that claim 74 is patentable over the cited references.

<u>Claim 90</u>

Independent Claim 90 is directed to an editor embodiment. The editor is used with a web browser and allows a user to edit a document being actively displayed by the browser "*wherein scripts contained in said document remain functional during editing.*" Further, the editor includes a client part, e.g., a first software program for execution within the browser that initiates editing functions when the user clicks on the displayed document.

The Examiner did not include specific reasoning for claim 90 but included claim 90 with his reasoning for claim 1. Because the Examiner has failed to provide detailed reasoning, he has not made a *prima facie* case for obviousness.

As explained in section A.ii above, WebWriter does not run the application during editing. In contrast, claim 90 requires scripts contained in the document to remain functional during editing.

In addition, claim 90 recites a first software program for execution within the browser. WebWriter I does not have such a program, since it does not seem to generate pages with client side scripts. WebWriter II does have client side scripts, however, as explained in section A.iv above, the combination of WebWriter I and WebWriter II is improper since they employ different architectures.

For all these reasons, applicant submits that claim 90 is patentable over the cited combination.

Claim 114

Independent Claim 114 is directed to an editor embodiment, but was rejected without any specific discussion in the grouping of claims described on page 3 of the recent Office Action. For this reason, applicant submits that the Examiner failed to make a prima facie case for obviousness.

Claim 114 is an independent claim describing a system for editing components on web document templates. Claim 114 requires "*the components including first features adapted to cooperate with an editor in dynamically editing said component.*" Claim 1 does not include such a feature, it does not even require "components". Therefore applicant submits that the examiners reasoning given is not applicable and therefore irrelevant for claim 114.

Claim 114 requires "*the components including first features adapted to **cooperate** with an editor*" (emphasis added). As reasoned in detail in section A.vii above, WebWriter does not teach or suggest components that cooperate with the editor. Applicant submits that therefore claim 114 is patentable over the cited combination.

Also, despite the Examiner's comments to the contrary in the advisory action a part of the editor cannot be interpreted as a component because the part lacks first features to cooperate with the editor in editing this part itself as required by the claim language "*components including first features to cooperate with an editor in editing said component.*" In addition, the use of the word cooperate makes clear that the components are not part of the editor.

<u>Claim 125</u>

Claim 125 is an independent claim describing a method useful for editing an application that is built using components and that operates by generating documents. The claim stands rejected as obvious based on the combination of WebWriter I and WebWriter II. In the Rejection, the Examiner did not include specific reasoning for claim 125 but handled the claim together with claim 1. For this reason, applicant submits that the Examiner failed to make a prima facie case for obviousness. Nevertheless claim 125 is a method claim while claim 1 is an apparatus claim. Therefore, applicant submits that the examiners reasoning given is not applicable for claim 125.

Claim 125 requires a method useful for editing an application that is built using components and that operates by generating documents. The first step is "*running the application, thereby executing selected components.*" As has been extensively explained in section A.ii and with regard to claim 1, neither of the cited WebWriter references perform this step during editing – they do not run the application during editing. Applicant submits that for these reasons, claim 125 is patentable over the cited combination.

C.   THE DEPENDENT CLAIMS ARE PATENTABLE OVER THE
      COMBINATION OF WEBWRITER I AND WEBWRITER II

      1.   Claims 2-5, 41 and 42

Claims 2-5, 41 and 42 are dependent from Claim 1 and should be considered patentable over the cited combination for all the same reasons.

With regard to claim 2, the Examiner cites page 2, column 1 of the WebWriter I article and refers to template pages, but does not explicitly describe what in WebWriter corresponds to the components. However, since the cited section introduces dynamic areas on the template pages, the examiner apparently interprets scripts called by dynamic areas of WebWriter as components. Claim 2 together with its base claim 1 requires a single page generator, which is generating editing features and executing components. The WebWriter Page Generator. however, is only running at run-time, after editing, and executing scripts, while the WebWriter Editor generates editing features during editing and not at run-time. Since the WebWriter Editor and the WebWriter Page Generator are two distinct programs running at different times, this is clearly distinct from a single page generator executing components and generating editing

features at the same time. Therefore, applicant submits claim 2 is patentable over the cited combination.

Claims 3-5 are dependent on Claim 2, and for all the same reasons, applicant submits that claim 3-5 are not taught or suggested by the cited prior art.

With regard to claim 3, the examiner cited Fig. 2 of WebWriter I. However, the cited figure just shows the user interface of the editor. It does not show any dynamic areas and consequently does not give any information on components. In contrast, claim 3 requires "*at least one of the components that reacts interactively on user input by executing instructions of said component on the server*". As reasoned in section A.ix, above, WebWriter does not teach or suggest anything like interactive components. For all these reasons, applicant submits that claim 3 is patentable over the cited combination.

Claims 4-5 are dependent on Claim 3, and for all the same reasons, applicant submits that claim 4-5 are not taught or suggested by the cited combination.

With regard to claim 4, the examiner cites WebWriter I Figs. 2 and 4. Fig. 4 shows a menu of available HTML tags, and does not give any information about dynamic areas. Fig. 2 also does not provide any information about dynamic areas. Claim 4, however, requires "*at least one of the components contains at least one other component*". Since components are apparently interpreted by the Patent Office as scripts associated with dynamic areas, both figures are then irrelevant for the showing one component containing another one. Section A.viii above explains that WebWriter does not teach or suggest one component containing another one. Therefore, claim 4 is patentable over the cited combination.

With regard to claim 5, the examiner cites page 9 of WebWriter I. However, page 9 reveals that each dynamic area is filled by its script exactly once per request. It is not possible that a script associated with a dynamic area is excluded from generation, or generated multiple times, and so the number of scripts of dynamic areas executed per template page is fixed. In contrast, the claim requires that the set of components on the generated documents can vary for different document requests. For all these reasons, applicant submits that claim 5 is patentable over the cited references.

With respect to claims 41 and 42, the examiner cites WebWriter II at page 1510, section 4.1, as disclosing a client part for execution on the client computer. Applicant refers to section A.iv above for a discussion of the propriety of the combination of WebWriter I and WebWriter

II. Specifically claim 41 read together with the base claim 1 implicitly requires the editor client part to work on the documents generated by the document generator program on the server ("an editor program dynamically operating on the generated documents"). Such a feature is not shown by WebWriter I, which works totally on the server side and does not have a client part. It is also not shown by WebWriter II, which includes parts that work on the client side. Thus, the combination of these different implementations is not readily obvious because it would require a significant effort to provide even just a basic compatibility between these two versions of WebWriter.

For all these reasons, applicant submits that claim 41 should be considered patentable over the cited combination. Claim 42 is dependent on Claim 41 and should be considered patentable over the cited combination for all the same reasons.

2.   Claim 8
Claims 8 depends from Claim 6, and for all the same reasons is patentable over the cited combination.

With regard to claim 8, the examiner cited Fig. 4 of WebWriter I, just as with claim 4. Both claim 4 and claim 8 require a component containing another component. Therefore, applicant submits that claim 8 is patentable over the cited combination because of the reasons discussed with regard to claim 4. Applicant also refers to the reasoning in section A.viii above showing that WebWriter does not teach or suggest nested components. In contrast, claim 8 explicitly requires that a component is nested within a component

3.   Claims 23-25
Claims 23-25 depend from Claim 22, and for all the same reasons, Claims 23-25 are not taught or suggested by WebWriter I, either alone or in combination with WebWriter II.

With regard to Claim 23, the examiner cited page 2 column 1 of WebWriter I and refers to use of the term "template page" as disclosing that the editor program operates functional applications in an edit mode. In applicant's claim, the word "applications" refers back to the base claim 22 and therefore means the applications being edited, not the editor itself. Claim 23 therefore requires the edited application be functional during editing by claiming "*the editor program operates functional applications in an edit mode permitting editing directly in the web browser*". Applicant also amended claim 23 to further emphasize that indeed the application

being edited is functional during editing by adding "operates a functional application" and "editing said application". As reasoned in section A.i above, page 2, column 1 of the WebWriter I article does not state that the edited application is running during editing ("*the static parts of each page are created in advance by the designer using a text editor*"). Section A.ii gives several citations to the WebWriter articles that support the premise that, in fact, the application is run only after editing and not during editing. During editing, dynamic parts are displayed as placeholders (See page 4, col. 2, last para.: "*WebWriter then adds a placeholder for the output area*"). For all these reasons, applicant submits that WebWriter does not teach or suggest "the editor program operates functional applications" as required by claim 23 and therefore, claim 23 is patentable over the cited combination.

Claim 24-25 depend from Claim 23, and for all the same reasons are patentable over the cited combination. With regard to Claim 24, the examiner cited pp. 2-9 of WebWriter I as disclosing a component that can react on subsequent document requests by executing selected instructions. Applicant submits that the scripts called by WebWriter's dynamic areas are just used for processing of a single request. As explained on page 9, left column, Heading The Web Writer Page Generator, the scripts are called during document generation to produce output for an output area during processing of a single request. The scripts do not have a concept of multiple requests or subsequent requests and can therefore also not react in any way on a subsequent request. In contrast, claim 24 requires at least one component that can react on subsequent document requests. Please also refer to section A.ix above. For all these reasons, applicant submits that claim 24 is patentable over the cited combination.

Claim 25 depends from Claim 24, and for all the same reasons is patentable over the cited combination. For Claim 25, the examiner cited WebWriter I pp. 2-9, et seq. as disclosing component classes. However, applicant submits that neither WebWriter I nor WebWriter II discloses component classes implementing components. The scripts associated with dynamic areas are unix scripts or programs as disclosed on WebWriter I page 9, second column first paragraph ("*runs the named program … The program is run in a separate Unix process*"). So the WebWriter Page Generator in fact calls unix scripts or programs. There is no teaching or suggestion in WebWriter that components are objects and implemented as classes.

In addition, WebWriter does not have a menu of components for insertion into the document templates. Figure 7(b) of the WebWriter I article shows forms for entering scripts, but not a menu.

Applicant submits that claim 25 is patentable over WebWriter I alone or in combination with WebWriter II since WebWriter does not teach or suggest component classes, a document generator program using component classes, nor a menu of components.

        4.   Claims 27-33, 43

Claims 27-33 and 43 depend from Claim 26, and for all the same reasons are patentable over the cited combination.

With regard to dependent claim 27, the examiner cites WebWriter I pages 2-9 et seq. as disclosing at least one component being executed by the first software program. Since base claim 26 requires the first software program to generate a second document having features which permit editing, the examiner must be interpreting the WebWriter Editor as the first software program. According to WebWriter I page 9 section "The WebWriter Page Generator" (see also section A.ii above), it is the WebWriter Page Generator executing scripts associated with dynamic areas (see also Section A.vi) and not the WebWriter Editor. In contrast, the claim requires the first software program to execute at least one component. Therefore, applicant submits that claim 27 is patentable over the cited combination.

With regard to claim 28, the examiner cites WebWriter I page 4 for "clicking on its handles" as disclosing that the second document includes handles and choosing one of the handles selects a component for an editing operation. According to the limitations of base claim 28, the second document is a document that already contains code generated by executing the components. So, the claim language indeed requires handles to occur in a document generated by executing components. In contrast, WebWriter shows handles during editing inside a document that contains placeholders. In addition, the cited portion of the WebWriter I article seems to deal with HTML elements rather than dynamic areas, which are discussed later in the article. For all these reasons, applicant submits that claim 28 is patentable over the cited combination.

Claim 29 depends from Claim 28, and for all the same reasons is patentable over the cited combination. With regard to claim 29, applicant submits that the delete operation is not described in the cited portion of the WebWriter I reference.

With regard to dependent claim 30, the examiner cites WebWriter I pages 2-9 et. seq. as disclosing that the features include scripts. However, applicant disagrees. The WebWriter Editor seems to send HTML documents, which do not contain scripts. In contrast, the claim implicitly requires the second document to contain scripts, since claim 30 requires the features to include scripts and base claim 26 requires the second document to include these features. Therefore, applicant submits that claim 30 is patentable over the cited combination.

Claim 31 depends from Claim 30, and for all the same reasons is patentable over the cited combination. With regard to dependent claim 31, the examiner cites WebWriter I pages 2-9 et. seq. as disclosing scripts that encapsulate information from the first document. As discussed with regard to claim 30, the WebWriter Editor does not work to include scripts into the generated documents. In contrast, the claim explicitly requires scripts that encapsulate information from the first document, and therefore applicant submits that claim 31 is patentable over the cited combination.

With regard to dependent claim 33, the examiner cites WebWriter II, page 1510 section 4.1 Applicant submits that a combination of WebWriter II and WebWriter I is improper as reasoned in section A.iv above, and for that reason, the citation of WebWriter II for the dependent claim is inadequate to teach or suggest the claimed invention.

In addition, the cited portion does not seem to reveal change requests. The WebWriter II seems to send a complete document to the server after editing on a save operation. On page 1511, 3rd paragraph "The CGI modules provide server-side services such as loading files, parsing HTML, **saving files** …" (emphasis added). In contrast to saving a complete document, claim 33 requires "*to send change requests for the first document to the server computer*", which mandates that a change be sent to the first document and not a complete document. For these reasons, applicant submits that claim 33 is patentable over the cited combination.

With regard to dependent claim 43, the examiner cites WebWriter II, page 1510 section 4.1 Applicant submits that a combination of WebWriter II and WebWriter I is improper as reasoned in section A.iv, and for that reason, this claim is considered patentable over the cited combination.

5.   Claims 52-58

Claims 52-58 depend from Claim 51, and for all the same reasons, these claims are patentable over the cited combination.

With respect to claim 53, the examiner cited WebWriter II page 1508 section 1.2, while he cited WebWriter I for the base claim 52 and 51. Applicant submits that for all the reasons given in section A.iv the combination of WebWriter I and WebWriter II is improper, and therefore the obviousness rejection should be withdrawn.

With respect to claim 54, the examiner cites WebWriter I pages 2-9 et. seq. as disclosing second documents that include HTML pages with embedded scripts. However, in the WebWriter I article, the WebWriter Editor does not use client side scripts. In contrast, the claim requires second documents to contain embedded scripts, and base claim 51 requires the second program to generate the second documents, and therefore applicant submits that claim 54 is patentable over the cited combination.

With regard to claim 55, the examiner cites WebWriter I pages 2-9 and refers to use of the term "editor" as disclosing edit functions include removing a component. The WebWriter I article does not appear to describe the function of deleting a component.

With respect to claim 57, the examiner cites WebWriter pages 2-9 and refers to the term "editor" as disclosing that the generated documents include edit features. The term generated document, however, refers back to claim 56, where it is used as the documents generated by the WebWriter Page Generator and by executing instructions of the components. As reasoned in section A.ii, however, the WebWriter Page Generator is separate from the WebWriter Editor and does not insert any editing features into the generated documents. In contrast, the claim specifically requires generated documents to include editing features. Therefore, applicant submits that claim 57 is patentable over the cited combination.

With respect to claim 58, the examiner cites WebWriter I pages 2-9 and refers to the term "editor" as disclosing instructions to allow the user to click on the generated documents to select items to perform edit functions on. The term generated documents, however, refers back to claim 56 , where it is used as the document generated by the WebWriter Page Generator and by executing instructions of the components. As reasoned in section A.ii, however, the WebWriter Page Generator is separate from the WebWriter Editor. Therefore it is not possible to click on a document generated by the WebWriter Page Generator to perform editing. In contrast, the claim

requires instructions to allow the user to click on the generated document to select items to perform edit functions on. Therefore, applicant submits that claim 58 is patentable over the cited combination.

6. Claims 60-73

Claims 60-73 are dependent from Claim 59, and for all the same reasons, applicant submits that Claims 60-73 are likewise not taught or suggested by WebWriter I, either alone or in combination with WebWriter II.

With regard to claim 60, the examiner cites section 4 of the WebWriter II article, while citing the WebWriter I article for base claim 59. Applicant submits that for all the reasons given in section A.iv above, the client side operations described in section 4 of WebWriter II cannot easily be combined with the server side functionality of WebWriter I. Therefore, the combination is improper and not effective to make the claims obvious.

Claims 61-63 depend from Claim 60, and for all the same reasons, these claims are patentable over the cited combination.

With regard to claim 61, the examiner cited WebWriter I page 2-9 as disclosing further instructions for execution during document generation to collect edit information for use by the editor. In fact, the WebWriter I article describes two kinds of document generation, the document generation done by the WebWriter Editor on page 8, and the document generation done by the WebWriter Page Generator on page 9. According to claim 59, "document generation" means "generating generated documents from dynamic web documents which look and function similar to end user's view of the documents". This is not the case for the document generation of the WebWriter Editor (see section A.ii). For the document generation of the WebWriter Page Generator, there is no teaching or suggestion of a feature "to collect edit-information" as required by the claim. Therefore, applicant submits that claim 61 is patentable over the cited combination.

With regard to Claim 63, the examiner cites WebWriter I page 2-9 for automatically repeating requesting that the document generator process the dynamic web document if required. As reasoned with regard to claim 59 and explained in section A.ii, the WebWriter Editor never does request that the WebWriter Page Generator generate a document, and so there is also not a repeating of this step in WebWriter I. In fact, the WebWriter Page Generator is only running at

run-time when the WebWriter Editor has already saved the document. Therefore, applicant submits that claim 63 is patentable over the cited combination.

With regard to Claim 66, the examiner cites page 2-9 of the WebWriter I article as disclosing the operations of inserting, deleting and modifying a component. However, the WebWriter I article does not describe the deletion of a component.

With regards to Claim 67, the examiner cites page 2-9 of WebWriter I as disclosing that the view looks, except for editing features, similar to the end-user view. In general, both views do not look similar as reasoned in section A.ii above. In fact, Figs. 7(b) and 6(b) of WebWriter I show both views, which clearly look different since the end user's view contains the output generated by scripts associated with dynamic areas while the view shown by the editor just shows placeholders. Also, see Figs. 10(b) and 11 for another example. In contrast, the claim requires both views to look similar except for editing features. Therefore, applicant submits that claim 67 is patentable over the cited combination.

With regard to claim 68, applicant refers to the reasoning for claim 61.

Claim 69 is dependent on 68, and for all the same reasons, applicant submits that Claim 69 is likewise not taught or suggested by WebWriter I, either alone or in combination with WebWriter II. With regard to claim 69, the examiner cited WebWriter I at pp. 2-9 as disclosing the editor using the edit information to modify the dynamic web document. However, in WebWriter I, the edit information is never collected as explained above with respect to claim 61. Therefore, applicant submits that claim 69 is patentable over the cited combination.

Claim 70 is dependent on 69, and for all the same reasons, applicant submits that Claim 70 is likewise not taught or suggested by WebWriter I, either alone or in combination with WebWriter II. With regard to claim 70, the examiner cites WebWriter I at pp. page 2-9. Claim 70 requires "*position information on the components contained in the document template*". Applicant submits that WebWriter I does not contain any teaching or suggestion of such position information. Applicant therefore submits that claim 70 is patentable over the cited combination.

With regard to claim 72, the examiner cited WebWriter I at pp. 2-9 as disclosing initiating a reload in the browser. A reload in the browser means a function of the browser to load the same page a **second time**. However, WebWriter I does not describe the editor as containing instructions for initiating a reload, as required in Claim 72. Therefore, applicant submits that claim 72 is patentable over the cited combination.

With regard to claim 73, the examiner cites WebWriter I at pp. 2-9. In contrast, claim 73 requires information on an element to be displayed without requesting the document generator. Applicant submits that WebWriter I does not have page generation in the WebWriter Editor when showing information on a dynamic area and associated script. There is no page generation in the WebWriter Page Generator since the WebWriter Page Generator runs only at run-time after editing. As explained in the reasoning of claim 59 and section A.ii, neither the WebWriter Editor nor the WebWriter Page Generator is a page generator in the sense of base claim 59 and therefore claim 73 should be patentable over the cited combination as well as claim 59.

7. Claims 75-89

Claims 75-89 depend from claim 74, and for all the same reasons, should be considered patentable over the cited combination.

With regard to claim 75, the examiner cites WebWriter I at pp. 2-9 and refers use of to the term "editor" as disclosing editing functions comprising adding, modifying and deleting a component. However, the function of deleting a component does not seem to be described in the cited document.

With regard to claim 76, the examiner cited WebWriter I at pp. 2-9 and refers to use of the term "templates" as disclosing components denoted on document templates using tag syntax, whereby the tag name identifies the component kind. However, WebWriter I on page 9 left column discloses that output areas with scripts are denoted using a tag of the form

formatting string

The tag name of this syntax is "output" while it is the script attribute that identifies what the script is to be called. In contrast, however, claim 76 explicitly requires the tag name to identify the component kind. Therefore, applicant submits that claim 76 is patentable over the cited combination.

With regards to claim 77, the examiner refers to a combination of WebWriter I and WebWriter II. Applicant refers to section A.iv and submits that a combination of WebWriter I and WebWriter II is improper, and should be withdrawn as a basis for rejection.

With regard to claim 78, the examiner cites WebWriter I at pp. 2-9 and refers to use of the terms "generator" and "templates" as disclosing that at least one component can be excluded from said generated document. However, the section "The Web Writer Page Generator" on page

9 makes clear that the WebWriter Page Generator processes each dynamic area on a template page by executing the associated script and has no ability to exclude a dynamic area from this process. In contrast, the claim requires "*at least one component that … can be excluded from said generated document upon selected document requests for said document template*".

In addition, the claim requires a component that can react interactively on subsequent document requests. As discussed in section A.ix, WebWriter does not seem to disclose interactive components. For all these reasons, applicant submits that claim 78 is patentable over the cited combination.

Claims 79-80 depend from Claim 78, and for all the same reasons, these claims are patentable over the cited combination. With regard to claim 79, the examiner cites WebWriter I at pages 2-9 as disclosing instructions to prevent excluded components from reacting on subsequent document requests. As discussed in section A.ix, scripts of dynamic areas of WebWriter cannot react on subsequent document requests and therefore WebWriter also does not include instructions to prevent this reacting.

Claim 80 depends from Claim 79, and for all the same reasons, this claim is patentable over the cited combination. With regard to claim 80, the examiner cites WebWriter I at pages 2-9 as disclosing storing information in session memory on some components that are present on the document generated. However, the section "The Web Writer Page Generator" on page 9 does not refer to session memory or storing of information in session memory on scripts of dynamic areas. The claim refers to actions taking place at run-time and therefore would have been described in the cited section if they had been present in WebWriter I. Thus, applicant submits that there is no teaching or suggestion for storing of information in session memory on scripts of dynamic areas in WebWriter and therefore the claim should be patentable over WebWriter.

With regard to claim 81, the examiner cites WebWriter I at pages 2-9 and refers to use of the term "template" as disclosing that at least one component can be excluded from said generated document. However, the section "The Web Writer Page Generator" on page 9 makes clear that the WebWriter Page Generator processes each dynamic area on a template page by executing the associated script and has no ability to exclude a script of a dynamic area from this process. In contrast, the claim requires "*instructions to decide about exclusions of components*".

In addition, claim 81 requires components nested inside the first component. Applicant refers to section A.viii for the discussion that WebWriter does not disclose nested components. For all these reasons, applicant submits that claim 81 is patentable over the cited combination.

With regard to claim 82, the examiner cites WebWriter I at pages 2-9 and refers to the term "display/interface" as disclosing an editor taking the varying set of components into account. However, the WebWriter Editor displays each dynamic area with associated script as exactly one placeholder. There is no mechanism in the WebWriter Editor to take a varying set of components into account. In contrast, the claim requires the editor be able to take the varying set of components into account. Also, as discussed with regard to claim 74, the WebWriter Page Generator cannot handle a varying set of components. For all these reasons, applicant submits that claim 82 is patentable over the cited combination.

With regard to claim 83, the examiner cited WebWriter I at pages 2-9 and refers to the terms "editor" and "template" as disclosing an editable view that includes and excludes selected components. However, the WebWriter Editor displays each dynamic area with associated script as exactly one placeholder. There does not seem to be a view that includes and excludes placeholders. In contrast, claim 83 requires a view that includes and excludes selected components. For all these reasons, applicant submits that claim 83 is patentable over the cited combination.

With regard to claim 84, the examiner cites WebWriter I at pages 2-9 and refers to the term "template" as disclosing a document generated for at least one document template that contains more components than the document template. In WebWriter, each dynamic area is filled by its script exactly once, as explained in section "The Web Writer Page Generator" on page 9. Therefore, a page generated by the WebWriter Page Generator cannot contain more components than the template page. In contrast, however, the claim requires that a generated document contains more components than the document template. Therefore, applicant submits that claim 84 is patentable over the cited combination.

With regard to claim 85, the examiner cites WebWriter I at pages 2-9 and refers to use of the term "template" as disclosing multiple instances of a component being included in a generated document. In WebWriter, each dynamic area is filled by its script exactly once, as explained in section "The Web Writer Page Generator" on page 9. In contrast, claim 85 requires

multiple instances. Therefore, applicant submits that claim 85 is patentable over the cited combination.

With regard to claim 86, the examiner cited WebWriter I at pages 2-9 as disclosing instructions to assign unique identifiers and to qualify names generated into the browser code with the unique identifiers. The cited portion, however, does not seem to reveal assigning unique identifiers or qualifying names. In particular, the section "The Web Writer Page Generator" on page 9, which describes the run-time page generation in the WebWriter Page Generator, does not describe this feature. Thus, applicant submits that claim 86 is patentable over the cited combination because the cited art does not seem to teach or suggest assigning of unique identifiers and qualifying names as required by the claim.

With regard to claim 87, the examiner cites WebWriter I at pages 2-9 and refers to use of the term "template" as disclosing instructions to decide how many instances of components are included in the documents generated. As reasoned with regard to claim 85, WebWriter does not teach or suggest multiple instances and therefore it is clear that it also does not have instructions to decide how many instances of components are included in the documents generated. In contrast, instructions to decide how many instances of components are included in the documents generated are specifically required by the claim. Therefore, applicant submits that claim 87 is patentable over the cited combination.

With regard to claim 88, the examiner cites WebWriter I at pages 2-9 and refers to use of the term "template" as disclosing an editable view that includes multiple instances. As reasoned with regard to claim 85, WebWriter does not teach or suggest multiple instances at run-time and therefore it is clear that does not have multiple instances in an editable view. Also, the description of the WebWriter Editor does not reveal multiple instances of placeholders or of scripts of dynamic areas. In contrast, the claim explicitly requires an editable view that includes multiple instances of selected components. Therefore, applicant submits that claim 88 is patentable over the cited combination.

With regard to claim 89, the examiner cites WebWriter I at pp. 2-9 and refers to use of the term "browser" as disclosing a sixth component including tenth instructions for displaying the sixth component during editing. Applicant submits that as reasoned in section A.ii the WebWriter Editor shows placeholders during editing and does not execute any instructions of the scripts associated with dynamic areas during editing. Therefore, applicant submits that

WebWriter does not disclose tenth instructions. In contrast, the claim requires a sixth component including tenth instructions for displaying the sixth component during editing. Therefore, applicant submits that claim 89 is patentable over the cited combination.

        8.   Claims 91-113

Claims 91-96 depend from Claim 90, and for all the same reasons are patentable over the cited combination.

With regard to claim 91, the examiner cited WebWriter I at pages 2-9 as disclosing a second browser window. In applicants reading, WebWriter I just uses a single browser window, for example, as shown in figure 1. In contrast, claim 91 explicitly requires a "second browser window". WebWriter II appears to use frames and not a second browser window. For all these reasons, applicant submits that claim 91 is patentable over the cited combination.

With regard to Claim 93, the examiner cites WebWriter I at pages 2-9 as disclosing that the original document looks similar to the document. Applicant submits that in general the original and the document shown during editing look clearly different in WebWriter since dynamic areas are replaced by placeholders. This is discussed in section A.ii. Therefore applicant submits that claim 93 is patentable over the cited combination.

Claims 94-95 depend from Claim 93, and for all the same reasons are patentable over the cited combination. With regard to Claim 94, the examiner cites WebWriter I at pages 2-9. The claim explicitly requires the original document to be a dynamic document containing components with instructions contained therein. As explained in section A.ii, in the case of dynamic documents, the WebWriter Editor displays a document that appears different than the original document since dynamic areas are replaced by placeholders. In contrast, base claim 93 requires that the original document and the document look similar. Therefore, applicant submits that claim 94 is patentable over the cited combination.

With regard to claim 96, the examiner cites WebWriter I at pages 2-9 as disclosing that links contained in the document stay functional during editing. The WebWriter I article does not include any teaching on how links are displayed in a document being edited. Applicant further submits that simply including a link in a document edited by the WebWriter Editor would leave the editor and navigate to the linked page. In contrast, applicant's claim requires that links stay

functional during editing in order to allow the user to browse and edit at the same time. For all these reasons, applicant submits that claim 96 is patentable over the cited combination.

<div align="center">

9.   Claims 115-124
</div>

Claims 115-124 and 128 depend from claim 114, and for all the same reasons, should be considered patentable over the cited combination.

With regard to claim 115, the examiner cited WebWriter I at pages 2-9 and refers to use of the term "editor" as disclosing that first features include instructions for passing information to the editor. According to base claim 114, components for execution on the server include first features for cooperation with the editor. Claim 115 states "*first features include fourth program instructions for passing information to the editor*" and depends on claim 114, which states that "*at least one of the components including first features.*" WebWriter does not describe that scripts called by WebWriters dynamic areas pass information to the editor. In fact, as discussed in section A.vi and section A.ii, these scripts are executed at run-time, after editing is finished, and so the scripts running later cannot pass information to the editor as required by the claim. Consequently, WebWriters scripts called by dynamic areas have no features for passing information to the editor. In contrast, claim 115 requires these features and therefore applicant submits that claim 115 should be patentable over the cited combination.

Claims 116-118 depend from Claim 115, and for all the same reasons, these claims are patentable over the cited combination.

With regard to claim 116, the examiner cited WebWriter I at pages 2-9 and refers to use of the term "editor" as disclosing that part of the information is collected during execution of the components on the server. However, the scripts called by WebWriter's dynamic areas do not collect information for the editor. As discussed in section A.vi and section A.ii, these scripts run at run-time after editing is finished and therefore cannot pass information to the editor. In contrast, claim 116 requires that part of the information is collected during execution of the components. Therefore, applicant submits that claim 116 should be patentable over the cited combination.

With regard to claim 117, the examiner refers to WebWriter II. Applicant submits, however, that "said information" is in fact never collected as reasoned with regard to base claim 115 and therefore also not transmitted either by WebWriter I or by WebWriter II.

With regard to claim 118, the examiner cited WebWriter I at pages 2-9 and refers to use of the term "editor" as disclosing that the information includes attributes of the component. As reasoned with regard to base claims 114 and 115, no information is transmitted from the running scripts associated with WebWriter's dynamic areas to the WebWriter Editor. In addition, the cited portion does not show that the scripts called by WebWriter's dynamic areas collect any attribute values to be passed to the editor. In contrast, base claims 114 and 115 require components to include fourth program instruction for passing information to the editor, and claim 118 requires the information to include attribute values of components. For these reasons, applicant submits that claim 118 is patentable over the cited combination.

With regard to claim 119, the examiner cited WebWriter I at pages 2-9 and refers to the use of term "editor" as disclosing first features including fifth instructions that display additional editing features. The base claim 114 requires "*components including first features*". In applicants reading, the cited portion does not reveal that scripts called by WebWriter's dynamic areas display any additional editing features. In fact, as reasoned in section A.vi and section A.ii, these scripts run at run-time only, i.e., after editing. Therefore, it makes no sense for them to display any editing features since editing is already finished. In contrast, claim 119 together with base claim 114 requires components including first features including instructions that display additional editing features. For these reasons, applicant submits that claim 119 is patentable over the cited combination.

Claim 120 depends from Claim 119, and for all the same reasons, this claim is patentable over the cited combination. With regard to claim 120, the examiner cites WebWriter I at pages 2-9 and refers to use of the term editor as disclosing that said editing features include handles. In applicants reading, handles are displayed by the WebWriter Editor. In contrast, claim 120 requires said editing features to include handles, base claim 119 requires first features include instructions that display additional editing features, and base claim 114 requires components including first features. So, the claim requires **components** to contain instructions to display handles. In contrast, in WebWriter it is the editor displaying the handles. For these reasons, applicant submits that claim 119 is patentable over the cited combination.

With regard to claim 121, the examiner cited WebWriter I at pages 2-9 and refers to the term editor as disclosing first features including extensions for use by the editor. Claim 121 recites "*first features include an extension for use by the editor*" and read together with the base

claim 114 "*at least one of the components including first features*" limits the extensions to be part of the components and not of the editor. The cited portion of WebWriter does not reveal any extensions for use by the editor being contained in the scripts called by WebWriter's dynamic areas. The scripts are used only at run time, when the editing is finished, as reasoned within sections A.vi and A.ii. Therefore, applicant submits that WebWriter does not teach extensions as required by the claim. For these reasons, applicant submits that claim 121 is patentable over the cited combination.

Claim 122 depends from Claim 121, and for all the same reasons, this claim is patentable over the cited combination. With regard to claim 122, the examiner cited WebWriter I at pages 2-9 and refers to use of the term "editor" as disclosing a web page for editing component attributes. Claim 122 explicitly requires "*said extension enables display of a page for editing the components attributes values.*" As reasoned with regard to base claim 121, the base claims require the extension to be part of components. In contrast, all the forms shown by the WebWriter Editor are built into the editor itself. For these reasons, applicant submits that claim 122 is patentable over the cited combination.

With regard to claim 123, the examiner cited WebWriter I at pages 2-9 and refers to use of the term "templates" as disclosing components denoted on document templates using tag syntax, whereby the tag name identifies the component kind. However, WebWriter I on page 9 left column discloses that output areas with scripts are denoted using a tag of the form

formatting string

The tag name of this syntax is "output" while it is the script attribute that identifies what the script is to be called. In contrast, however, claim 123 explicitly requires the tag name to identify the component kind. Therefore applicant submits that claim 123 is patentable over the cited combination.

With regard to claim 124, the examiner cited WebWriter I at pages 2-9 and refers to use of the term "browser" as disclosing second program instructions to generate browser code for displaying the component during editing. According to the base claim 114, second program instructions are part of selected components. As reasoned within section A.vi and section A.ii, WebWriter's scripts associated with dynamic areas are not executed during editing. The WebWriter Editor just displays a placeholder for scripts of dynamic areas during editing. In contrast, claim 124 requires second instructions in the components for displaying the

components during editing. Therefore, applicant submits that claim 124 is patentable over the cited combination.

>   10.   Claims 126-127

Claims 126-127 depend from Claim 125, and for all the same reasons are patentable over the cited combination.

With regard to claim 126, the examiner cited WebWriter I at pages 2-9 as disclosing repeating the running and the displaying steps after applying a modification function. In applicants reading, as reasoned in section A.ii, WebWriter runs the application only after editing is finished and the document is saved. So there is no running step during editing and consequently also no repeating of the running step in WebWriter. In contrast, the claim requires repeating the running and the displaying steps after applying a modification function. Therefore, applicant submits that claim 126 is patentable over the cited combination.

Claims 127 depends on Claim 126, and for all the same reasons is patentable over the cited combination.

II. CONCLUSION

For all the foregoing reasons, applicant submits that the claims are in condition for allowance, and the Examiner's reconsideration to that end is respectfully solicited. The Examiner is encouraged to telephone the undersigned should additional issues remain.

Respectfully submitted,

Date: _____December 30, 2008_____   By: _____/Richard A. Nebb/_____
                                            Richard A. Nebb
                                            Reg. No. 33,540

VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105
Telephone: 415.369.9660
Facsimile: 415.369.9665